

Projet de Génie Logiciel : Tests pour la sélection des astronautes de l'ESA



Damien SCHUMACHER
Marie-Eugénie MECHAIN

ENSC
16/12/2016

TABLE DES MATIERES

Introduction.....	2
I) Gestion de projet.....	2
a) UML.....	2
b) Planning	3
c) Maquettes.....	4
II) Spécifications détaillées	5
a) Les classes	5
b) Forms	6
c) Justification des choix de conception	10
III) Tests et résultats	11
a) Résultats	11
b) Tests	12
Tests unitaires	12
Tests fonctionnels	12
IV) Perspectives	14
a) Bilan	14
b) Améliorations possibles	14
Conclusion	15

INTRODUCTION

Le projet de génie logiciel qui nous est proposé cette année a pour but de créer une application permettant de tester les candidats se présentant à l'ESA pour devenir astronaute. Cette application devait être créée sous la forme d'un Windows Form. Dans ce rapport nous exposerons la démarche de gestion de projet et nous expliquerons notre code.

Il s'inscrit dans l'unité d'enseignement de deuxième année « **Génie Logiciel** » et fait partie du module « **Sciences fondamentales** » dispensé à l'École Nationale Supérieure de Cognitique.

I) GESTION DE PROJET

a) UML

Au tout début de notre réflexion par rapport au projet, nous avons pensé à faire une classe « Exercice » qui aurait été la classe mère de chaque test. Cependant nous avons décidé d'abandonner cette idée au fur et à mesure de l'avancée du projet, car nous avons de moins en moins de temps et que cela ne nous paraissait pas être le plus important. Nous avons donc 5 classes principales représentant chacune un test et étant reliées à un Windows Form.

Nous avons aussi beaucoup discuté quant à la mise en place d'un seul ou de plusieurs Windows Forms. Nous avons finalement choisi de mettre plusieurs Windows Forms. Chacun est relié d'un côté au menu principal et de l'autre à un test. Nous trouvons plus intuitif de cliquer sur un bouton qui nous ouvrirait une nouvelle page comprenant l'exercice, plutôt que de faire apparaître sur cette page et effacer les données du menu principal.

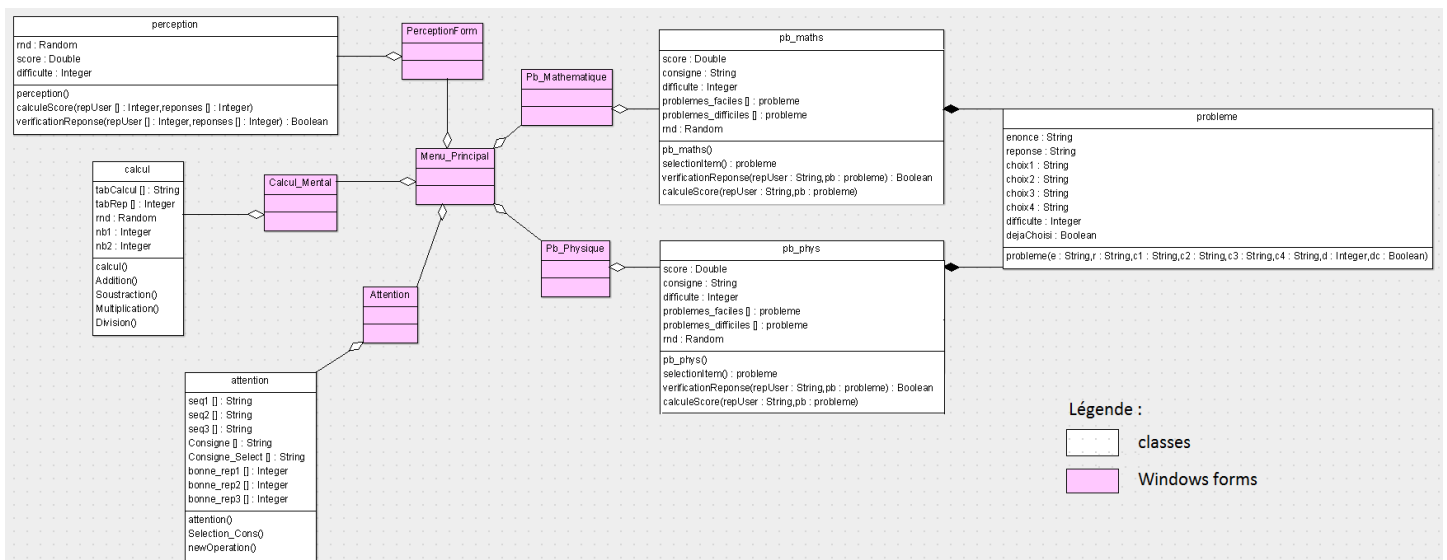
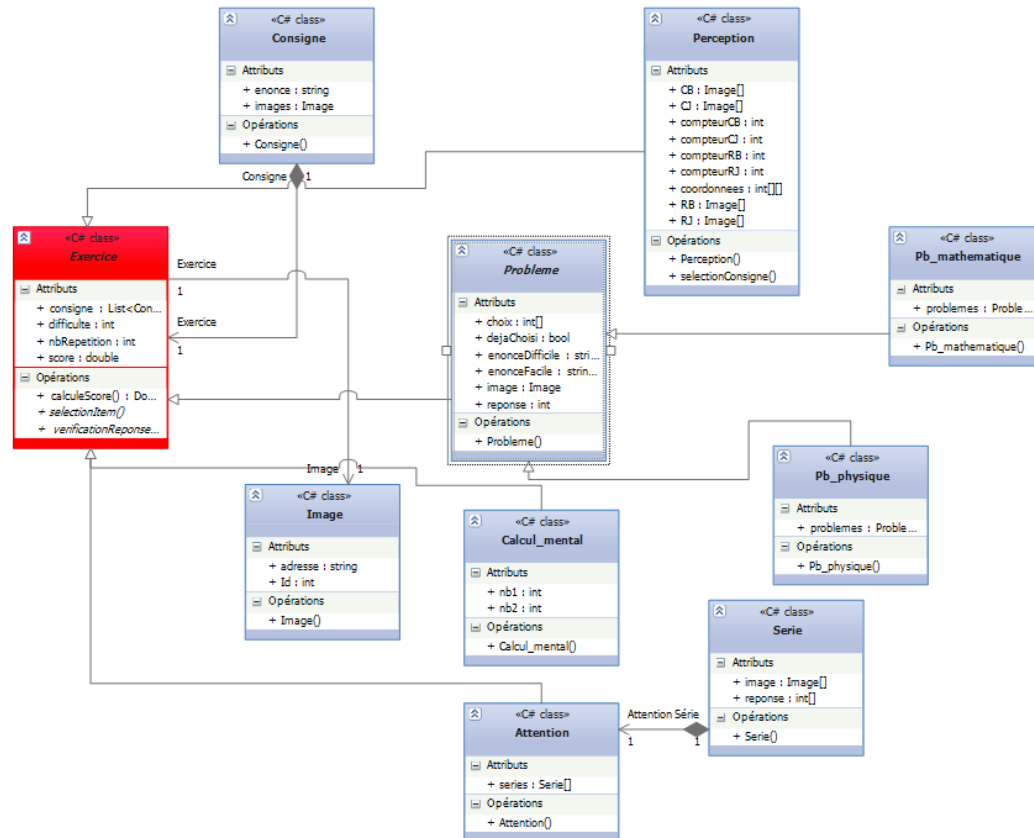
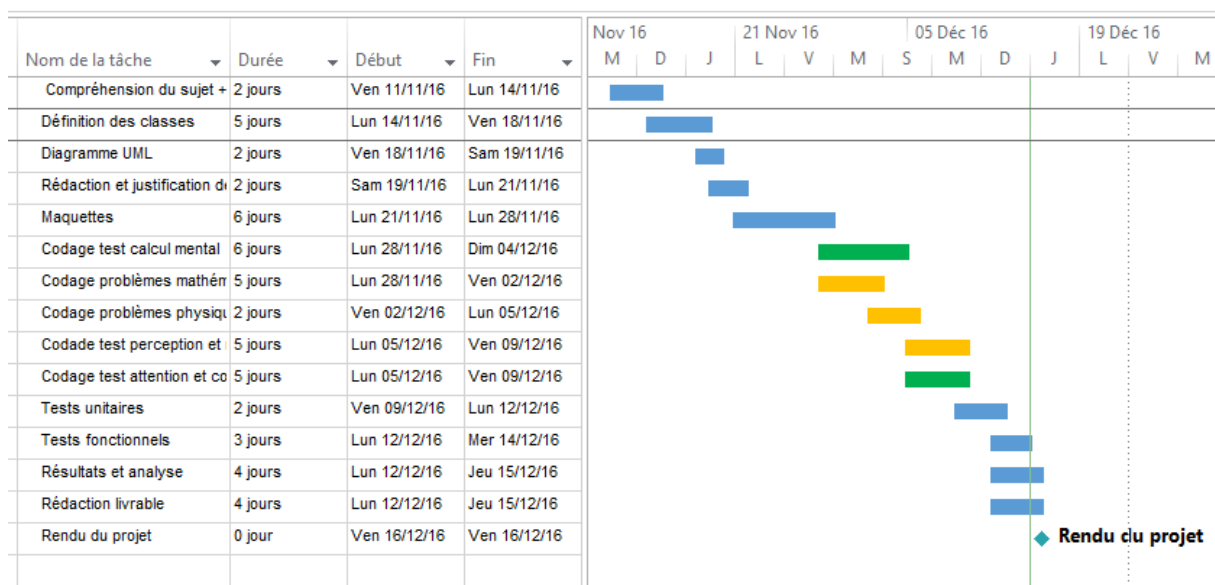


Diagramme des classes actuel



Ancienne version du diagramme des classes

b) PLANNING

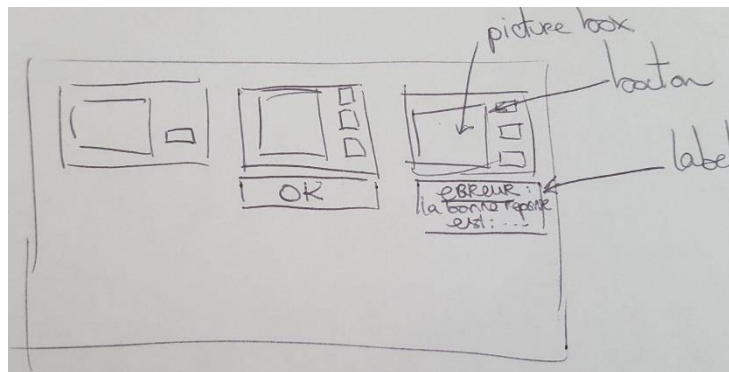


Ce planning a été édité au début du projet afin de nous fixer un temps défini pour chaque tâche que nous avions à faire. En bleu on peut voir les tâches effectuées en binôme, en orange celles effectuées par Marie-Eugénie et en vert celles effectuées par Damien. Nous avons essayé de nous répartir le travail de manière équitable. Nous avons respecté ce planning au début, mais le codage a pris plus de temps que ce que l'on pensait, nous avons donc perdu du temps sur cette étape, ce qui fait que nous n'avons pas eu le temps de faire le mode difficile de l'exercice « Attention et concentration ».

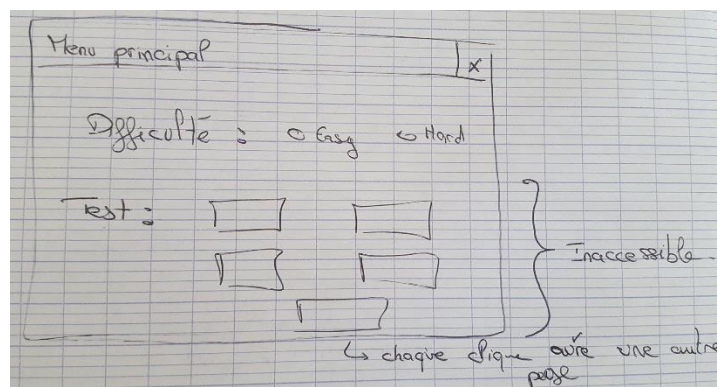
c) MAQUETTES

Au début du projet, nous nous sommes mis d'accord sur les choix à effectuer au niveau des Forms en fonction des besoins pour l'interface avec l'utilisateur.

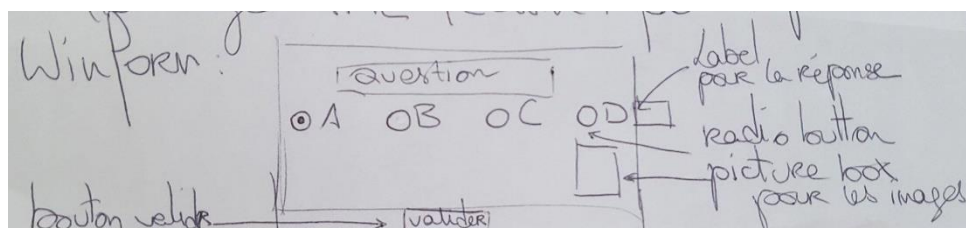
Pour cela nous avons fait à la main des maquettes très simples de tous les tests. On verra par la suite que ces maquettes ont bien été respectées, malgré quelques petites modifications. Ces modifications vous seront expliquées par la suite. Vous trouverez ci-dessous nos maquettes.



Maquette papier pour le test " Attention et concentration "



Maquette papier pour le menu principal



Maquette papier des tests " problèmes "

II) SPÉCIFICATIONS DÉTAILLÉES

a) LES CLASSES

calcul :

La classe calcul_mental possède comme attributs les deux nombres avec lesquels on fait un calcul. Elle instancie deux tableaux, le premier comprend les équations, et le second les réponses à ces équations.

Les deux nombres sont additionnés, soustraits, multipliés ou divisés en fonction du type de calcul sélectionné. Ainsi on retrouve dans le premier tableau l'équation (par exemple : « 1 + 1 ») et dans le second la réponse (dans l'exemple précédent : 2).

pb_maths :

La classe pb_maths possède comme attributs : score (double), consigne (string), difficulte (int), problemes_faciles et problemes_difficiles (tableaux de probleme) et rnd (Random).

Cette classe possède une méthode selectionItem qui sélectionne aléatoirement et retourne un problème en fonction de la difficulté de l'exercice qu'a choisie l'utilisateur. Elle possède aussi deux méthodes permettant respectivement de calculer le score de l'utilisateur et de vérifier ses réponses. Ces deux méthodes prennent en entrée la réponse de l'utilisateur et le problème auquel il a répondu. La méthode calculeScore, qui retourne le score de l'utilisateur utilise la méthode verificationReponse, qui elle retourne un booléen en fonction de la justesse de la réponse de l'utilisateur.

pb_phys :

La classe pb_phys possède les mêmes arguments et les mêmes méthodes que la classe pb_maths car les deux exercices sont quasiment les mêmes. Cependant, comme il n'y a pas de niveau de difficulté dans le test « problèmes physiques », cette classe possède un seul tableau de problèmes nommé problemes.

probleme :

La classe probleme sert à remplir les tableaux de problèmes des tests « problèmes physiques » et « problèmes mathématiques ».

Ses attributs sont : enonce (string), reponse (string), choix1 (string), choix2 (string), choix3 (string), choix4 (string), difficulte (int) et dejaChoisi (bool).

perception :

La classe perception possède comme attributs : rnd (Random), score (double), difficulte (int). Le score et la difficulté sont initialisés à 0, comme dans les classes pb_maths et pb_phys.

Toujours comme dans les classes pb_maths et pb_phys, cette classe possède les méthodes calculeScore et verificationReponse. Cependant, comme dans ce test l'utilisateur doit fournir trois réponses à chaque mémorisation, ces méthodes prennent en argument un tableau de réponses utilisateur ainsi qu'un tableau contenant les bonnes réponses de l'exercice.

attention :

La classe attention possède les attributs suivants : seq1 (tableau de string), seq2 (tableau de string), seq3 (tableau de string), Consigne (tableau de string), Consigne_Select (tableau de string), bonne_rep1 (tableau d'int), bonne_rep2 (tableau d'int), bonne_rep3 (tableau d'int).

Elle possède une méthode permettant de sélectionner une consigne en fonction de la série d'images proposée et d'ajuster les bonnes réponses de l'exercice en fonction.


b) FORMS

Présentation globale :

Menu Principal

Bienvenue dans l'application de tests pour la sélection des astronautes de l'ESA !
Vous avez ici la possibilité d'effectuer cinq tests et de choisir leur difficulté.

Vous pouvez revenir au menu principal à tout moment pour effectuer un autre test ou le même test avec une difficulté différente.
Pour cela, il suffit de cliquer sur la croix rouge située dans le coin haut droit de la fenêtre de test.



DIFFICULTÉ ☐ Facile ☐ Difficile

TESTS

Perception et mémoire associative

Attention et concentration

Calcul mental

Problèmes mathématiques

Problèmes physiques

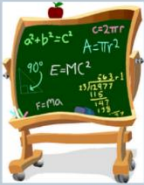
Perception et mémoire associative

Memorisez puis retranscrivez les chiffres présents dans les ronds jaunes.
Attention, la position des formes est importante.

A	B	C	D
6	5	3	7
E	F	G	H
2	3	4	6
I	J	K	L
8	6	2	1

Problème mathématique

Répondez aux problèmes posés.




La calculatrice de Léo est vraiment en mauvais état. Les seules touches en état de marche sont 5 7 + =.

Quel nombre ne peut-il pas afficher ?

☒ 10 ☐ 12 ☐ 14 ☐ 16

Attention et Concentration



Bouton 1

Bouton 2

Bouton 3

Form1

Quel type de calcul souhaitez-vous effectuer ?

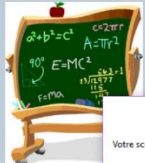
Addition Soustraction Multiplication Division

fin de l'exercice

L'épreuve est terminée !
Vous avez 0% de bonnes réponses

Problème physique

Répondez aux problèmes posés.



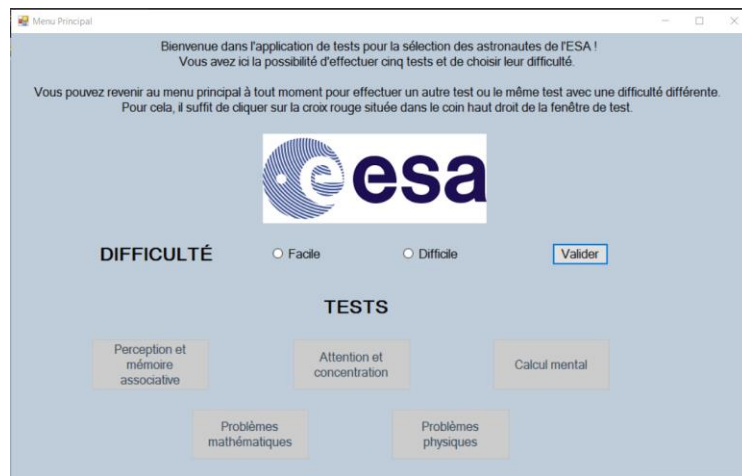
Dans un circuit électrique, le courant I sachant :

☒ 40 A ☐ 80 A ☐ 22 A ☐ 10 A

Votre score est de 4/10.

On peut voir sur la représentation ci-dessus que nous avons 6 interfaces qui seront expliquées par la suite. Pour les tests « problèmes mathématiques », « calcul mental » et « perception », il est possible de choisir la difficulté que le candidat désire avoir.

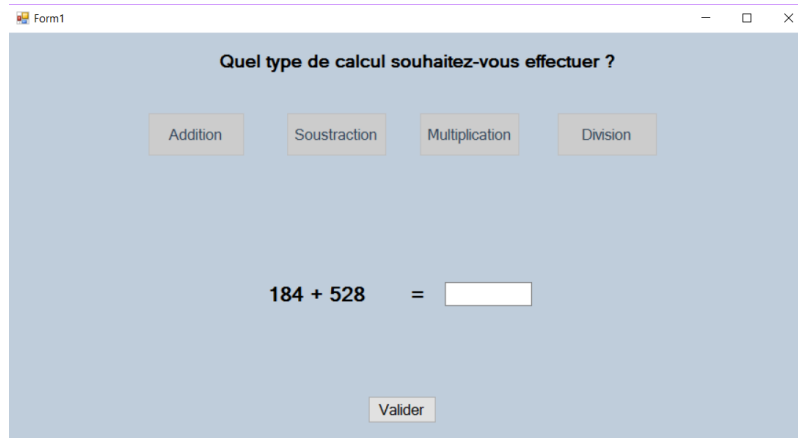
Menu principal :



Le menu principal est le centre de notre application. C'est lui qui permet de se rendre sur chaque page de test, mais c'est aussi la seule page accessible lorsqu'on est sur la page d'un test.

Comme vous pouvez le voir il est composé d'une phrase d'accueil, de radio buttons permettant de choisir la difficulté de l'exercice et de cinq boutons. Chaque bouton renvoie à une page de test.

Calcul mental :



La page calcul mental permet, comme son nom l'indique, d'évaluer le niveau de calcul de l'utilisateur. Durant un test, le candidat est soumis à 10 calculs qu'il doit faire de tête. À l'issue de ces 10 calculs il reçoit son pourcentage de bonnes réponses.

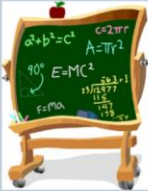
Au départ le candidat doit choisir la difficulté de l'exercice. Ensuite il décide du type de calcul qu'il veut effectuer. Il a le choix entre addition, soustraction, multiplication ou division. Une fois qu'il a choisi, une équation et une TextBox apparaissent.

En mode facile, le candidat a tout le temps qu'il veut pour répondre et appuyer sur le bouton valider. Une fois qu'il a appuyé, un label apparaît et indique à l'utilisateur si sa réponse est correcte ou non. Si elle ne l'est pas l'utilisateur voit aussi la bonne réponse. En mode difficile, le candidat a 5 secondes pour répondre : s'il n'a pas appuyé sur le bouton valider au bout des 5 secondes, l'équation et la TextBox s'enlèvent et un message apparaît.

Problèmes mathématiques :

Problème mathématique

Répondez aux problèmes posés.



La calculatrice de Léo est vraiment en mauvais état. Les seules touches en état de marche sont 5 7 + =.

Quel nombre ne peut-il pas afficher ?

☒ 10
 ☐ 12
 ☐ 14
 ☐ 16

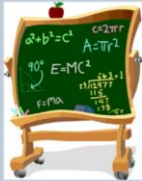
Valider

La page problèmes mathématiques affiche la consigne de l'exercice en haut de la fenêtre. En dessous, une PictureBox permet d'afficher l'image correspondant au problème s'il y en a une, une image neutre sinon. En dessous de cette image s'affichent l'énoncé du problème, ainsi que les quatre choix de réponses possibles sous forme de radio button. Enfin, tout en bas de la fenêtre, un bouton valider est disponible pour que l'utilisateur puisse valider sa réponse. En cas de mauvaise réponse, un label « mauvaise réponse » s'affiche pendant un temps assez court à côté du bouton valider. En cas de bonne réponse, c'est le même principe. Seule la valeur du label change. En fin d'exercice, une message box apparaît pour indiquer le score (sur 10) de l'utilisateur.

Problèmes physiques :

Problème physique

Répondez aux problèmes posés.



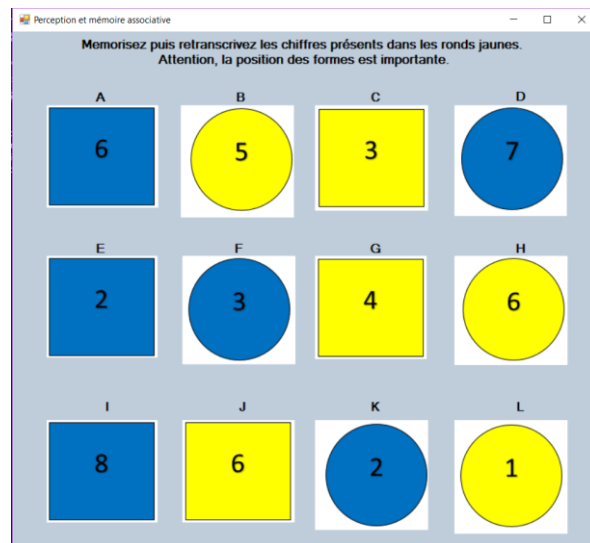
Lorsqu'une voiture roule à 100 km/h, la résistance de l'air est proportionnelle à ?

☒ Sa masse
 ☐ Sa vitesse
 ☐ Au carré de sa vitesse
 ☐ Sa masse et sa vitesse

Valider

La page problèmes physiques est construite de la même façon que la page problèmes mathématiques.

Perception :



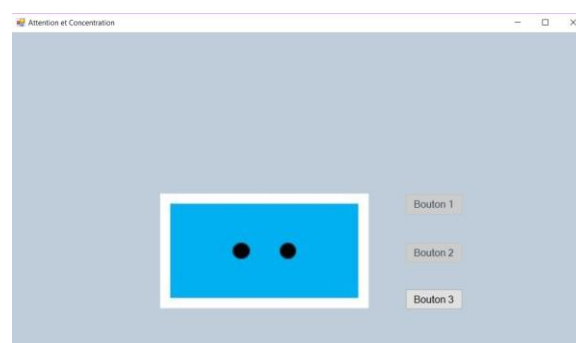
Quand l'utilisateur a sélectionné l'exercice perception dans le menu principal, une message box apparaît et affiche la consigne du test. Le test ne débute qu'une fois que l'utilisateur ferme cette message box.

La page perception affiche la consigne de l'exercice tout en haut de la page. En dessous de cette consigne, 12 PictureBox sont présentes pour pouvoir afficher les 12 formes à mémoriser. Ces dernières ne s'affichent que durant 2 ou 4 secondes (selon la difficulté choisie) grâce à un timer.

Une fois le temps d'affichage écoulé, les composants du formulaire sont cachés afin de laisser apparaître le formulaire de réponse. Nous avons choisi cette méthode car la transmission d'informations est plus aisée ainsi. Le formulaire de réponse est composé de trois TextBox ayant pour labels les lettres correspondant aux positions des formes à mémoriser en fonction de la consigne. La consigne reste aussi affichée dans ce formulaire. Il dispose enfin d'un bouton valider pour que l'utilisateur puisse soumettre ses réponses. Si l'utilisateur appuie sur le bouton valider sans avoir rempli tous les champs du formulaire, une message box apparaît et lui indique de remplir tous les champs.

En fin d'exercice, une message box apparaît et donne le pourcentage de bonnes réponses de l'utilisateur.

Attention :



Dans attention le candidat va voir 3 séries d'images. Ces images sont des formes (rectangle, rond ou triangle) de couleur (bleu, rouge, vert, jaune) avec un certain nombre de points à l'intérieur (allant de 0 à 4 points).

Au début on affiche la consigne de l'exercice au candidat, la consigne est la suivante : « Lors de cet exercice vous allez voir 3 séquences de formes (rectangle, cercle ou triangle) de différentes couleurs (rouge, bleu, jaune, vert) avec des points à l'intérieur (de 0 à 4 points). À côté de chaque image vous aurez

3 boutons (sauf pour la première). Chaque bouton a une signification. Les deux premiers expriment une caractéristique de l'image en présence, en fonction de l'image précédente (par exemple, le bouton 1 exprime que les deux images ont la même forme). Le troisième est là pour tous les autres cas. »

Après qu'il ait appuyé sur le bouton « OK » les significations du bouton 1 et 2 sont affichées. Quand le candidat appui sur « Prêt » les significations s'effacent et une forme apparaît ainsi que les 3 boutons. Pour la première image de chaque série, seul le bouton 3 est accessible. Après, à côté de chaque image les trois boutons sont accessibles. Il faut appuyer sur un bouton en fonction d'un point commun ou non de l'image avec l'image précédente. À la fin de chaque série les significations des boutons réapparaissent. Lors de la fin de la dernière série, une MessageBox apparaît avec le pourcentage de bonnes réponses. En mode facile, le candidat a tout le temps qu'il veut pour répondre et les significations des boutons ne changent jamais. Nous n'avons pas réussi à coder le mode difficile.

c) JUSTIFICATION DES CHOIX DE CONCEPTION

Dans notre programme, nous avons utilisé en grande partie la convention de nommage camelCase. La seule variation que nous lui avons imposée est que les noms des classes ne commencent pas par une majuscule, mais par une minuscule.

Nous avons de plus décidé de coder en français, car c'est notre langue maternelle et c'est donc un réflexe pour nous d'utiliser des mots français.

Enfin, nous avons décidé de passer à la ligne après chaque accolade ouvrante et fermante et d'indenter le code afin d'obtenir une meilleure lisibilité.

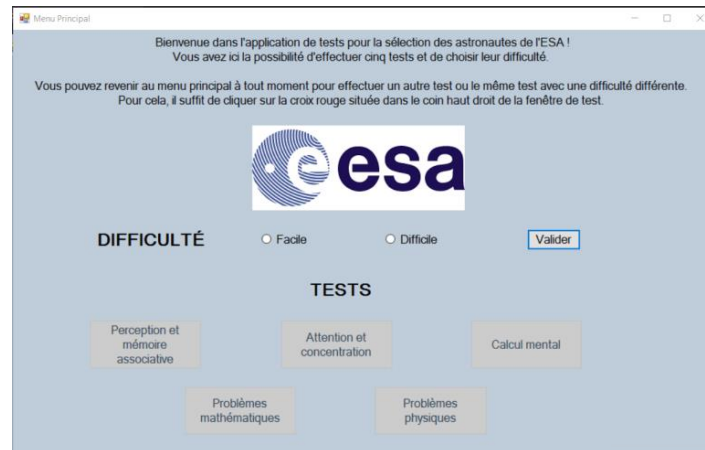
Pour ce qui est de l'architecture du logiciel, nous avons choisi une architecture en séparation des responsabilités. En effet, nous avons créé un espace de noms pour chaque test et pour le menu principal. Il était ainsi plus aisé pour notre groupe de se répartir le travail et le travail d'un membre du groupe n'influaient pas sur le travail de l'autre membre.

Enfin, pour ce qui est du design de l'interface, nous avons choisi de mettre le fond des formulaires en bleu car le logo de l'ESA est bleu. Nous avons choisi de mettre une couleur bleu sur le mouse over des boutons pour que l'utilisateur repère mieux le bouton sur lequel il a mis son curseur de souris. Enfin, nous avons choisi la même police d'écriture dans tous les formulaires (Microsoft sans serif, taille 12) pour ne pas perdre l'utilisateur quand il change de formulaire et pour que tous les textes soient bien lisibles.

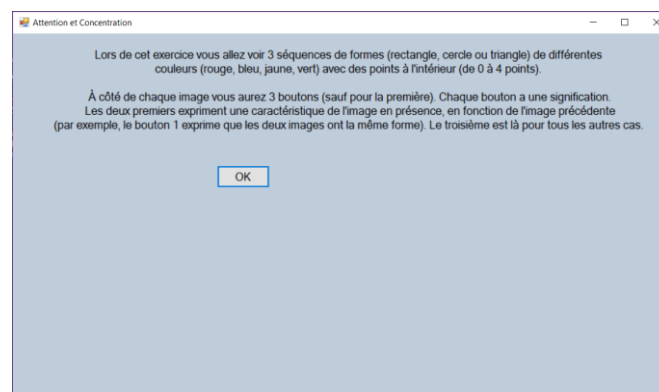
III) TESTS ET RÉSULTATS

a) RÉSULTATS

Le principe de l'application est expliqué à l'utilisateur dans le menu principal.

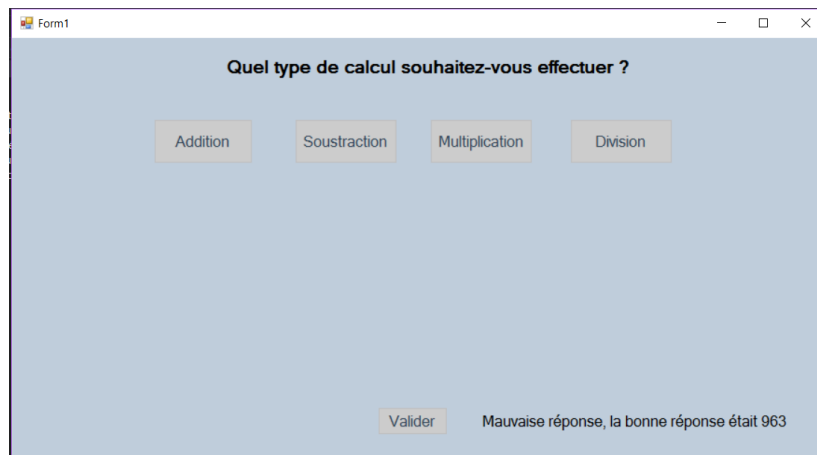


Avant le début d'un exercice, la consigne de ce dernier est affichée à l'utilisateur, ainsi qu'un bouton pour qu'il puisse débiter l'exercice quand il le souhaite.

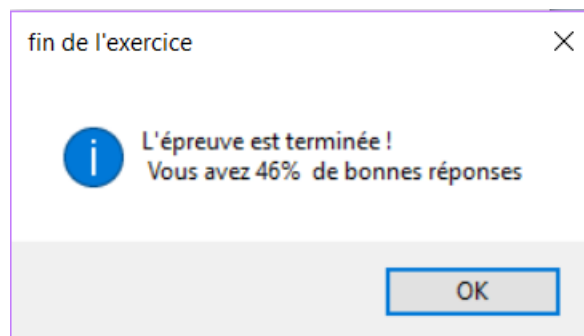


Dans certains exercices, des formulaires de réponses sont affichés à l'utilisateur pour qu'il saisisse ses réponses.

Durant l'exercice, un label affiche à l'utilisateur s'il a bien répondu ou s'il a mal répondu à la question posée. Le label lui indique aussi quelle était la bonne réponse dans certains cas.



A la fin de chaque exercice, les scores sont annoncés par une messageBox. Ils sont soit par points sur 10 ou soit par pourcentage de bonnes réponses.



b) TESTS

Tests unitaires

Malheureusement nous n'avons pas eu le temps de réaliser des tests unitaires. De plus, nous manquons de pratique à ce niveau là.

Si nous avions eu plus de temps et de pratique, nous aurions testé des méthodes telles que « selectionItem » ou « calculeScore ». Nous aurions aussi pu tester les méthodes d'opérations de la classe calcul.

Tests fonctionnels

Protocole de tests :

L'application est déjà lancée. L'utilisateur est face au menu principal.

Etape 1 : Choisir une difficulté.

Etape 2 : Choisir l'exercice « Perception et mémoire associative ».

Etape 3 : Remplir le formulaire de réponses et passer à la mémorisation suivante.

Etape 4 : Finir la série.

Etape 5 : Revenir au menu principal.

Etape 6 : Choisir un autre test ainsi que sa difficulté.

Nous n'avons pas eu le temps de réaliser ces tests fonctionnels sur des personnes extérieures au groupe, mais nous les avons réalisés nous-mêmes. Ils peuvent révéler certaines faiblesses de notre interface : revenir au menu principal à la fin d'un exercice n'est pas toujours évident, même si l'instruction est expliquée dans le menu principal, l'utilisateur peut être bloqué au moment du choix de test s'il n'appuie pas sur le bouton valider pour la difficulté.

IV) PERSPECTIVES

a) BILAN

En bilan de ce projet, nous l'avons trouvé très fastidieux. En effet, le nombre d'exercices à coder étant important et le nombre de TP consacrés au code du projet étant très restreint, nous avons été pris par le temps.

De plus, le projet faisait appel à de nombreuses techniques que nous n'avions pas eu l'occasion de manipuler auparavant.

Par exemple, l'utilisation de Git et GitHub était encore très floue pour nous au début du projet. Nous avons mis du temps à prendre ces outils en main, temps non consacrée à la réflexion à l'architecture et au code du logiciel.

Spécifiquement pour notre groupe, nous n'avions jamais fait de sérialisation XML auparavant. Nous avons pensé remplir nos classes à l'aide de fichiers XML au début du projet, nous avons réalisé ces XML mais au final nous n'avions pas de méthode de sérialisation viable et compréhensible pour nous donc nous avons abandonné la sérialisation.

Enfin, le sujet de ce projet était tout de même intéressant et le projet constituait un bon entraînement à la gestion de projet, à la réflexion de l'architecture de code et au codage d'un logiciel Winform complet.

b) AMÉLIORATIONS POSSIBLES

Avec plus de temps, nous aurions pu terminer la partie difficile du test « Attention et Concentration » et ajouter des problèmes dans la base de données des problèmes physiques et mathématiques.

Nous aurions pu faire une sérialisation XML correcte. Nous aurions aussi pu choisir d'utiliser une base de données à la place, car nous avons vu comment gérer une base de données en première année.

Nous aurions aussi pu mettre un peu plus de design dans notre interface pour la rendre plus agréable. En effet, nous nous sommes principalement intéressés à ses fonctionnalités.

Il est possible de s'imaginer avoir des séries qui soient vraiment aléatoires, car dans notre application nous avons décidé de les prédéfinir. Pour cela il faudrait reprendre notre code. L'application pourrait évoluer en laissant la possibilité aux administrateurs d'ajouter, ou retirer un exercice.

CONCLUSION

Ce projet nous a apporté une approche concrète de ce qu'est le génie logiciel. On a pu voir que la phase de préparation du codage est très importante. Elle permet de ne pas partir dans tous les sens. On a aussi remarqué qu'il ne faut pas sous-estimer le temps à accorder au codage car c'est principalement cela qui nous a fait défaut.